ESC: Energy Synchronized Communication in Sustainable Sensor Networks

Yu Gu, Ting Zhu and Tian He
{yugu,tzhu,tianhe}@cs.umn.edu
Department of Computer Science and Engineering, University of Minnesota

***Abstract*—With advances in energy harvesting techniques, it is now feasible to build sustainable sensor networks (SSN) to support long-term applications. Unlike battery-powered sensor networks, the objective of sustainable sensor networks is to effectively utilize a continuous stream of ambient energy. Instead of pushing the limits of energy conservation, we are aiming at energy-synchronized designs[1] to keep energy supplies and demands in balance. Specifically, this work presents the Energy Synchronized Communication (ESC) as a transparent middleware between the network layer and data link layer that controls the amount and timing of RF activity at receiving nodes. In this work, we first derive a delay model for cross-traffic at individual nodes, which reveals an interesting *stair effect* in low-duty-cycle networks. This effect allows us to design a localized energy synchronization control with $\mathcal{O}(1)$ time complexity that *shuffles* or *adjusts* the working schedule of a node to optimize cross-traffic delays in the presence of changing duty-cycle budgets. Under different rates of energy fluctuations, shuffle-based and adjustment-based methods have different influences on *logical connectivity* and *cross-traffic delay*, due to the inconsistent views of working schedules among neighboring nodes before schedule updates. We study the tradeoff between them and propose methods to update working schedules efficiently. To evaluate our work, ESC is implemented on MicaZ nodes with two state-of-the-art routing protocols. Both test-bed experiment and large scale simulation results show significant performance improvements over randomized synchronization controls.**

I. Introduction

With the increasing need for cyber-physical interaction, Wireless Sensor Networks (WSN) have emerged as a key technology for many long-term applications, such as military surveillance [1], [2], field monitoring [3] and assisted living [4]. Due to the stringent constraints on cost and form factors, traditional battery-powered sensor networks must balance the tradeoff between sustainability and system performance [5]–[7]. Normally, the design objective of these systems is to *conserve as much energy as possible* while meeting minimal requirements [8]–[10].

Because sensor networks usually interact with the physical environment, they are especially well-suited to exploit ambient energy resources. For example, there are already many existing technologies to extract energy from the environment such as solar, thermal, optical, and kinetic energies [11]–[13]. In addition, several recent works have built prototypes [14]–[16] to demonstrate the feasibility of deploying *sustainable sensor network* with energy-harvesting nodes. However, influenced by the traditional belief in energy management, the designers of those systems still think that *maximum* energy harvesting and *minimum* energy consumption are always beneficial.

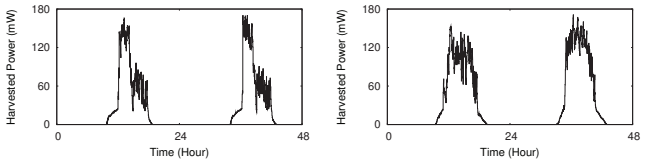
Different from previously ingrained belief, we take a new position in this work. We argue that energy management should *synchronize* the supply with demand. The equilibrium point is achieved when the energy supplied and demanded are in balance. It is not always beneficial to conserve energy when a network can harvest excessive energy from the environment since energy storage devices (e.g., batteries or capacitors) are always limited in capacity and usually leakage-prone. Therefore, energy saving with reduced performance during energy-rich periods is actually wasteful and counter-productive. In other words, in sustainable sensor networks, we would like to *consume as much energy as possible* while maintaining their sustainability.

In this work, we are particularly interested in studying the impact of energy synchronization on communication performance. We propose *Energy Synchronized Communication* (ESC), a novel solution that *dynamically synchronizes node activity patterns with available energy budgets, so as to minimize communication delay at individual nodes* in sustainable sensor networks. Specifically, by exploiting an interesting stair-effect of delay during energy synchronization, ESC is capable of minimizing communication delay at individual nodes in constant time and lies as a generic middleware between the data link layer and network layer. The major intellectual contributions of this work are as follows:

- An Energy Synchronized Communication (ESC) protocol is designed as a generic middleware service for supporting existing network protocols. To our best knowledge, this is the first in-depth and generic work to study low duty-cycle communication performance in energy-harvesting, sustainable sensor networks.
- We formulate a general model for cross-traffic delay at individual nodes. This model reveals an interesting *stair effect of delays* in low duty-cycle networks. We show that, counterintuitively, the communication delay is not affected by when a packet is received as long as the packet arrives within a certain interval. This stair effect allows us to design a localized $\mathcal{O}(1)$ algorithm to minimize communication delay while synchronizing duty-cycles of individual nodes with available ambient energy.

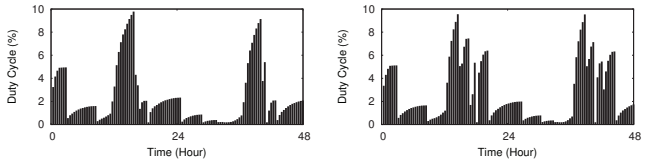
[1]Usually synchronization refers to time dimension, for energy-harvesting sensor networks we propose to use the term synchronization to represent the balance between energy supply and demand in the network.

978-1-4244-4634-6/09/$25.00 ©2009 IEEE



(a) Node 4 (b) Node 5

Fig. 1. Harvested Power



(a) Node 4 (b) Node 5

Fig. 2. Duty Cycle

- *Logical Link Quality (LLQ)* is put forward to reflect the true reliability of a link in low-duty-cycle networks, capturing transient inconsistency views among neighboring nodes. We study the impact of *shuffle-based* and *adjustment-based* synchronization on LLQ and propose how to maintain LLQ with a small overhead.

The rest of the paper is organized as follows: Section II motivates the necessity of ESC. Section III articulates the network model and related assumptions. Section IV formalizes a delay model for cross traffic. Section V and Section VI describe our energy-synchronized control algorithm and the impact of updates on logical link quality. Section VII discusses practical design issues. Section VIII and Section IX present test-bed and simulation evaluation results, respectively. Section X briefly discusses related work. Section XI concludes the paper.

II. Motivation

The motivation of this work is originated from our empirical experience of deploying energy-harvesting, sustainable sensor networks. In such networks, the energy supply usually is very dynamic and the whole network normally has to operate at low-duty-cycle to ensure the sustainability. With those unique characteristics, effective data communication in such networks therefore poses a new challenge beyond traditional static, battery-powered sensor networks. In the following paragraphs, we explain the impact of dynamic energy supply and low-duty-cycle networking on data communication process in detail.

Fig. 3. Experiment Site

First, in energy harvesting networks, the harvested energy usually is very unpredictable and could vary significantly over time [15], [16]. To study the empirical energy harvesting rate over time, we have deployed 11 solar-powered sensor nodes in an open field, shown in Figure 3. We collect the energy harvesting rates for the 11 deployed sensor nodes for a period of two days and Figure 1 plots the harvested energy over time for node 4 and node 5 in Figure 3. Clearly in Figure 1, in the time dimension, the harvested energy varies significantly both within a day and across days. Furthermore, in the space dimension, even node 4 and node 5 are physically co-located, their energy harvesting rates also vary significantly. Furthermore, although energy-harvesting sensor nodes usually are equipped with rechargeable batteries or super capacitors to alleviate the impact of energy variations, they are limited in energy storage capacity due to the small form factor requirement and waste energy unnecessarily due to the problem of energy leakage [16]. With such energy dynamics in the network, we can no longer schedule sensor nodes *a priori* as most existing battery-powered solutions do. Consequently, the question of how to effectively synchronize energy supply with sensor node activities while optimizing communication process, introduces a very demanding task.

Second, in the long run, ambient energy (e.g., solar, wind) is normally not intensive enough to sustain the continuous 100% duty cycle operation of sensor nodes [15]–[17]. Figure 2 shows the affordable duty cycles of node 4 and node 5, given the energy harvested in Figure 1. From Figure 2, we can see in order to ensure the continuous operation, the duty cycles of an energy-harvesting node can only range from 0.2% to 9.78%. From our empirical measurement results, even in a sunny day, the total energy harvested at a node can only supply itself to operate at 100% duty cycle for 6.37 hours. Essentially, during the operation of an energy-harvesting network, sensor nodes have to activate very briefly and stay in a dormant state for a very long period of time. In order to forward a packet in such always-dormant networks, a sender would experience *sleep latency* - the time spent waiting for the receiver to wake up [18]. Moreover, since communication links between low-power sensor devices are usually unreliable [19], it brings further challenges in managing communication in sustainable sensor networks.

To the best of our knowledge, no prior work has studied the impact of *energy synchronization* on *low-duty-cycle networks with unreliable links*. We contribute this research direction with (i) theory, (ii) architecture and (iii) design.

III. System Models and Assumptions

Before presenting ESC in detail, we introduce the models and assumptions used in this work. In addition, we elaborate on the packet delivery process and define sleep latency in low-duty-cycle sensor networks. To simplify our description, we introduce our ESC design assuming (i) neighboring nodes are synchronized in the unit of a time instance and, (ii) there is at most one packet transmission within such a time instance. Later on in Section VII, we discuss how we can relax those assumptions in practice.

A. Working Schedule

The working schedule of a sensor node denotes the active-dormant behaviors of the sensor node over its lifetime. It consists of a set of *active instances*, during which a node can receive packets. Each active instance j at node i can be represented by a tuple (t_j^i, d_j^i), where t_j^i denotes the starting time of the active instance and d_j^i denotes the corresponding duration of the active instance j. Since many sensor node working schedules are periodic [20]–[22], it is sufficient to represent an infinite sequence of active instances, using repeated subsequences with a period time T. Let Γ_i be the working schedule of node i and the number of active instances within a period be N, we can have $\Gamma_i = \{(t_1^i, d_1^i), (t_2^i, d_2^i), ..., (t_N^i, d_N^i)\}$. According to its working schedule, a node continuously transits its state between active and dormant state. The duty cycle of node i, therefore is $\frac{N}{T}$. For example, Figure 4 shows a periodic working schedule $\Gamma_i = \{(1,1),(5,2),(8,1)\}$ with a period time 10. The duty cycle of node i here is $\frac{3}{10} = 30\%$.

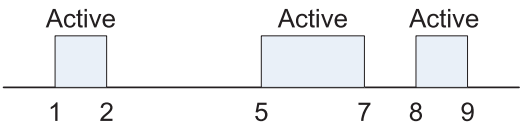


Fig. 4. A Working Schedule

To simplify our description, in the rest of the paper we assume all active instances have the same durations (τ). When a node is said to be active at time t, it has an active instance that starts at time t with duration of τ. We note that this definition of working schedule can actually accommodate active instances with varying durations. Essentially, if we let τ be the finest granularity of time durations in the design, we can represent any node schedule with the fixed τ.

B. Network Model

We assume a network with N sensor nodes. At a given point of time t, a sensor node is in either an active or a dormant state. When a node is in the active state, it can receive packets transmitted from neighboring nodes. When a node is in the dormant state, it turns off all function modules except a timer (for the purpose of waking itself up). In other words, a node can *wake up to transmit a packet at any time, but can receive packets only when it is in its active state*. Since a node can only receive packets during its active state, the *packet ready time* at a node (i.e., the time a node receives a new packet and ready to forward to the next hop node), therefore *is the same as active instances in its working schedule*.

For a sensor node, its neighbors consistently transit between active and dormant states, and thus connectivity between a pair of nodes varies over time and becomes time-dependent. Formally, for a given time t, we denote the network topology as $G(t) = (V, E(t))$, where V is a set of N nodes within the network, and $E(t)$ is a set of directed edges at time t. An edge e(i,j) belongs to $E(t)$ if and only if (i) node i is within the communication range of node j, and (ii) node j is active and hence able to receive packet at time t. Essentially $G(t)$ represents the potential traffic flow within the network at time t.

C. Sleep Latency in Low-Duty-Cycle Network

In connected networks, a one-hop packet delivery latency usually includes processing delay, transmission delay, and propagation delay, which are normally in the order of milliseconds. In low-duty-cycle sensor networks, however, a sender may need to wait for its receiver to wake up before it can send a packet. We define *sleep latency* as the time duration from the moment a packet is ready at the sender to the moment the destined one-hop receiver is active. Sleep latency is usually in the order of seconds, which is much longer than other delivery latencies. Therefore, in this paper we only consider sleep latency for measuring E2E delay. In a network with perfect links, the E2E delay is the sum of sleep latencies along the path of data delivery.

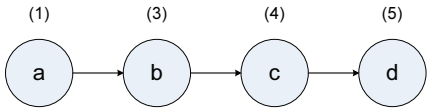


Fig. 5. A Linear Network

To further illustrate the concept of sleep latency, we use the linear network shown in Figure 5 as an example. In Figure 5, the active instance of each node is labeled on top of each node. Assume node a has a packet ready to be sent at time 1, given that the node b wakes up at time 3, the sleep latency for the first attempted transmission from node a to node b therefore is $3 - 1 = 2$.

IV. Modeling of Cross-Traffic Delay

This section presents the model for cross-traffic delay at a node. In low duty-cycle networks, a packet is transmitted only when the receiver is in the active state, and nodes in such networks only activate very briefly and stay in the dormant state for the majority of time. Thus here we assume data traffic/congestion is low, which holds well for existing low-duty-cycle sensor networks [18], [21], [23].

A. Cross-Traffic Pattern

ESC is designed to be a flexible middleware between the network layer and data link layer, so that it can be used to support various existing routing protocols. Therefore, it is important to have a delay model that can capture the behavior of cross-traffic (many-to-many), a generalized case for one-to-one, many-to-one, and one-to-many traffic.

For a node b in the network, depending on the specific routing protocol adopted, there may be a set of nodes that forwards packets to node b, and we call those nodes *predecessors* of node b. Similarly, for different final destinations or multi-path routing protocols, node b would forward its packets to a certain set of nodes, and we call those nodes *successors* of node b. For example, in Figure 6, the predecessors of node b include node p_1, p_2 and p_3, while the successors of node b are nodes s_1 and s_2. Here we note that a node can be both a

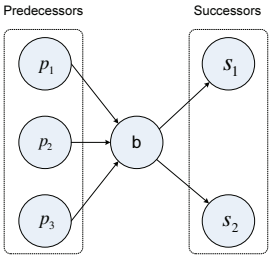


Fig. 6. Predecessors and Successors

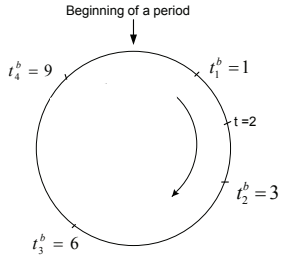


Fig. 7. A Cyclic Working Schedule

predecessor and a successor of node b, if this node exchanges data bidirectionally with node b.

To model the cross-traffic delay related to node b, we consider the expected delays for packets from all predecessor nodes through node b, then to corresponding successor nodes.

B. Delay Modeling

Assume at a predecessor node p_1, a packet destined to node b is ready at time t, where $t \in [0, T)$. Since the radio link between a pair of nodes is usually not perfect, node p_1 may need to initiate multiple transmissions before the packet has successfully arrived at node b. In order to obtain a corresponding sleep latency for each attempted transmission at node p_1 after packet ready time t, we introduce a cycle representation of a node working schedule shown in Figure 7.

In Figure 7, the cycle is equally divided by T ticks. Beginning at the 12 o'clock position, the time increases from 0 to T clockwise. Consequently, we can easily label the sequence of active instances at node b on the cycle. To measure the sleep latency (denoted as $L_t^b(k)$) introduced by node b, for a given k^{th} attempted transmission at time t, we can simply start from time t, follow the clockwise direction and measure the total distance traversed by visiting k labels after time t on the cycle. For example, as depicted in Figure 7 where T is 10 units of time, the packet is ready at time $t = 2$ and node b wakes up during time 1, 3, 6 and 9. For the first attempted transmission ($L_t^b(1)$), the corresponding sleep latency is 1 since the total time traversed for visiting one label (t_2^b) from time 2 is 1. Similarly, for the fourth attempted transmission ($L_t^b(4)$), the sleep latency is total time traversed from t to t_2^b, then to t_3^b, t_4^b and the fourth label t_1^b, which gives $1 + 3 + 3 + 2 = 9$ in total.

Link Reliability: Let the bi-directional link quality p_{ab} denote the success ratio of a round-trip transmission (DATA and ACK) between node a and node b. The probability that packet transmission succeeds at the k^{th} attempt is the probability that previous $k-1$ attempts failed times the probability that the current attempt succeeds, which is simply the link quality p_{ab}. Therefore the probability that the packet reaches node b from node a at its k^{th} attempt can be expressed as $(1-p_{ab})^{k-1}p_{ab}$.

Assuming the maximum number of packet retransmissions within the network is R_{max}, for the packets arriving at node b from a, the probability that they arrive at k^{th} attempt is under the condition that the packet is delivered within R_{max} retransmissions. The probability that a packet is delivered within R_{max} retransmissions can be expressed as $1-(1-p_{ab})^{R_{max}}$, and the corresponding conditional probability can be written as:

$$P^{ab}(k) = \frac{(1-p_{ab})^{k-1}p_{ab}}{1-(1-p_{ab})^{R_{max}}} \quad (1)$$

Delay Over a Single link: For a packet ready time t at node a, the expected transmission delay to reach node b is the sum of the product of probability that the packet reaches node b at its k^{th} attempt and corresponding sleep latency. Consequently, it can be formulated as:

$$D_{ab}(t) = \sum_{k=1}^{R_{max}} P^{ab}(k) L_t^b(k) \quad (2)$$

Delay from One Predecessor to One Successors: For a packet ready time t at a predecessor node p_i, assuming the packet arrives at node b at the k^{th} attempted transmission, its corresponding delay is simply $L_t^b(k)$. Upon receiving the packet at time $t + L_t^b(k)$, node b would forward the received packet to a destined successor node s_j with a delay of $D_{bs_j}(t + L_t^b(k))$. Then the expected delay from predecessor p_i to successor s_j with packet ready time t at predecessor p_i is the product of probability that the packet arrives node b at its k^{th} attempted transmission and corresponding cross-traffic delay, which can be expressed as:

$$D_{p_i s_j}(t) = \sum_{k=1}^{R_{max}} P^{p_i b}(k)(L_t^b(k) + D_{bs_j}(t + L_t^b(k))) \quad (3)$$

Delay from Multiple Predecessors to Multiple Successors: To model the expected cross-traffic delay at node b from all packet ready times at all predecessor nodes to all successor nodes, node b needs to know the portion of traffic that reaches node b from each packet ready time at all predecessor nodes to all successor nodes. To obtain those statistics, each predecessor of node b can piggyback packet ready time for each sent packet. Then at node b, with known sender and packet ready time for each packet, it can easily keep track of what percentage of packets is from a given packet ready time t at a predecessor node p_i, to a specific successor node s_j. Assume the number of packet ready time at a predecessor node p_i is N_{p_i}, for each packet ready time $t_k^{p_i}$, we can represent the percentage of traffic from a packet ready time $t_k^{p_i}$ at a predecessor node p_i to a successor node s_j as $W_k^{p_i s_j}$. Let the number of predecessor nodes and successor nodes at node b be N^p and N^s, respectively, we can express the expected delay of cross-traffic at node b as:

$$D_b = \sum_{i=1}^{N^p} \sum_{j=1}^{N^s} \sum_{k=1}^{N_{p_i}} W_k^{p_i s_j} D_{p_i s_j}(t_k^{p_i}) \quad (4)$$

V. Energy Synchronization Control

With the cross-traffic delay model available, we now introduce energy synchronization control for minimizing communication delay in sustainable sensor networks. In this section, we first assume the working schedules of predecessors and successors of a node are known and up-to-date. Later in Section VI, we discuss the impact of obsolete schedules and methods to keep the schedules up-to-date.

A. Main Idea

As shown in Section II, energy harvested from surrounding environments varies significantly over time at a sensor node. In order to make full use of the available energy supply, we need to enhance system performance when there is abundant scavenged energy available; conversely, we need to decrease duty-cycles with a minimum performance degradation when there is a shortage in the power supply. Previous works [24], [25] have demonstrated methods for deciding appropriate duty-cycle of a sensor node with in-situ energy supply. In this section, we further focus on the impact of duty-cycle changes on communication delay. More specifically, when we increase the duty-cycle of a node with additional available energy, we aim at minimizing cross-traffic delay at the node. Similarly, when a node needs to decrease its duty-cycle due to lack of sufficient energy supply, we would like to achieve a minimum increase in the cross-traffic delay of the node. As a result, harvested energy at an individual node is synchronized to minimize the communication delay within the network.

B. Decrementing Single Active Instance

For a given node b, assume its current working schedule is $\Gamma_b = \{t_1^b, t_2^b, ..., t_{N_b}^b\}$ and current energy supply can only afford $N_b - 1$ active instances to guarantee the sustainability of the node. Therefore, we need to remove one active instance from node b's current working schedule such that the increase of cross-traffic delay at node b is minimized. Since sustainable sensor nodes work in extremely low duty-cycles, the number of active instances in its working schedule is very small and would always below a constant value c. Consequently, to find the optimal active instance for decrement, we can simply test all existing active instances and select the one that yields the minimum delay at node b. The complexity of the optimal single active instance decrement, therefore, is just $\mathcal{O}(1)$.

C. Augmenting Single Active Instance

Intuitively, for augmenting one active instance at a node, we can perform an exhaustive search on all time instances within a period time T and choose the time instance that yields a minimum cross-traffic delay at a node. Although this naive algorithm is acceptable, a much more efficient algorithm can be designed based on the stair effect as presented in this section.

For a given node b, we can divide its period time into multiple intervals according to active instances of its predecessors and successors. For example, as shown in Figure 8, assume the predecessor and successor of node b is node p and node s, respectively. Let the active instances at node p and node s be $\{t_1^p, t_2^p\}$ and $\{t_1^s, t_2^s\}$, respectively. According to their locations on the cyclic working schedule, we can easily obtain four intervals, namely: (t_1^p, t_1^s), (t_1^s, t_2^p), (t_2^p, t_2^s) and (t_2^s, t_1^p).

Intuitively, one would expect the timing of an augmented active instance within an interval to yield different cross-traffic delays at node b. However this is not the case. We observe that *given schedules of predecessors and successors of node b, cross-traffic delay at node b only depends on the* **counts**, *instead of* **timing**, *of active instances within each interval.*

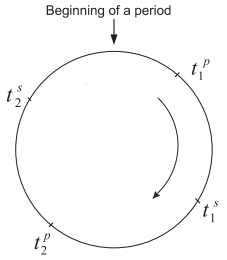


Fig. 8. Period Partition Example

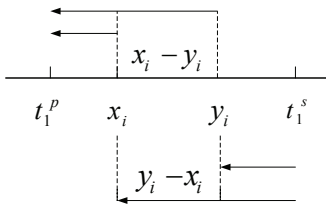


Fig. 9. Delay Difference Example

To validate this observation, it is sufficient to prove that for two arbitrary layouts of active instances of same size within an interval, the cross-traffic delays at node b are the same if packets are received within this interval. Following is the formal proof.

Lemma 5.1: let $X = x_1, x_2, ..., x_n$ and $Y = y_1, y_2, ..., y_n$ be two sets of active instances of node b within one interval (where $x_1 < x_2 < ... < x_n$ and $y_1 < y_2 < ... < y_n$), and let D^X and D^Y be corresponding cross-traffic delays. **If** $|X| = |Y|$, $D^X = D^Y$.

Proof: For any packet ready time t at a predecessor node p, since $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_n$ are in the same interval, as shown in Figure 9, the difference of sleep latencies for the i^{th} attempted transmission for X and Y after time t is just the difference of their respective active instances. Consequently, $L_t^X(i) - L_t^Y(i) = x_i - y_i$, for any $1 \le i \le n$.

Similarly, since x_i and y_i are in the same interval, the j^{th} attempted transmission from either x_i or y_i reaches the same active instance at a successor node s. Clearly shown in Figure 9, $L_{x_i}^s(j) - L_{y_i}^s(j) = y_i - x_i$, where $1 \le i \le n$ and $1 \le j \le R_{max}$.

By applying Equation 3 to X and Y, we have,

$$\begin{aligned} & D_{ps}^X(t) - D_{ps}^Y(t) \\ &= \textstyle\sum_{i=1}^{R_{max}} P^{pb}(i) \\ &(L_t^X(i) - L_t^Y(i) + D_{bs}(L_t^X(i)) - D_{bs}(L_t^Y(i))) \\ &= \textstyle\sum_{i=1}^{R_{max}} P^{pb}(i)(x_i - y_i + D_{bs}(x_i) - D_{bs}(y_i)) \end{aligned}$$

Since,

$$\begin{aligned} D_{bs}(x_i) - D_{bs}(y_i) &= \sum_{j=1}^{R_{max}} P^{bs}(j)(L_{x_i}^s(j) - L_{y_i}^s(j)) \\ &= \sum_{j=1}^{R_{max}} P^{bs}(j)(y_i - x_i) \end{aligned}$$

As $\sum_{j=1}^{R_{max}} P^{bs}(j) = 1$ we have:

$$D_{bs}(x_i) - D_{bs}(y_i) = y_i - x_i$$

Consequently,

$$D_{ps}^X(t) - D_{ps}^Y(t) = \textstyle\sum_{i=1}^{R_{max}} P^{pb}(i)(x_i - y_i + y_i - x_i) = 0$$

As a linear combination of $D_{ps}^{X}(t) - D_{ps}^{Y}(t)$ for all packet ready times at all predecessor nodes, we have $D^X - D^Y = 0$. ■

According to Lemma 5.1, for augmenting one active instance within a partitioned time interval, the cross-delay at node b is not affected by the timing of the augmented active instance. In other words, single active instance augmentation yields the same cross-traffic delay at node b within each of partitioned time intervals. Consequently, for a single active instance augmentation, we have a *stair effect of cross-traffic delays at node* b within each partitioned time interval. Based on this counterintuitive observation, we can have the following theorem on finding the optimal single active instance augmentation in the constant time.

Theorem 5.2: Assuming there are c intervals partitioned by active instances at predecessors and successors of a node and x_i is a random active instance within interval i, where $i = 1, 2, ..., c$. Let D^j represent the cross-traffic delay at the node after augmenting an active instance j to its original working schedule, and $Min(D^{x_i}) = D^{x_k}$, where $i = 1, 2, ..., c$, then any time instance within interval k is the optimal single active instance augmentation and the complexity of this process is $\mathcal{O}(1)$.

Proof: Within any time interval i that is partitioned by active instances at predecessors and successors, for any random augmented active instance x_i, essentially we increase the number of active instances within the interval by one. By Lemma 5.1, it is straightforward that those augmented active instances yield the same cross-traffic delays at node b. Therefore, to find the optimal augmented active instance at a node, we just need to check the cross-traffic delay with a random active instance augmentation within each of time intervals, and find the interval k that yields the minimum cross-traffic delay.

Since sustainable sensor networks operate in extremely low duty-cycle, we would have a very few packet ready times from predecessor nodes and active instances from successor nodes. Consequently, there would be only a few time intervals that need to be checked in order to find the optimal augmented active instance. In other words, the number of time intervals c is a constant value, and thus the complexity of finding optimal active instance is just $\mathcal{O}(1)$. ■

Guided by Theorem 5.2, we can derive the detailed energy synchronization process at a node b shown in Algorithm 1. First, based on working schedules of predecessor and successor nodes of node b, we create c time intervals and initialize system variables (Line 1 to Line3). Then for each time interval, we select a random time instance x_i, augment it to working schedule of node b and calculate cross-traffic delay D_b (Line 5 to Line 7). For each calculated D_b, we compare it to current minimal cross-traffic delay D_{min} and store new minimal delay value and the corresponding time interval if necessary (Line 8 to Line 11). After testing each time interval, we obtain the optimal active instance augmentation for node b. Since time complexity of our energy synchronization process is just $\mathcal{O}(1)$,

Algorithm 1 Active Instance Augmentation at a Node b

Input: Working schedule Γ for predecessor and successor nodes
Input: Real-time link quality for cross-traffic from predecessor and successor nodes
Input: Traffic distribution for cross-traffic
Output: The optimal augmented active instance

1: Sort active instances for predecessor and successor nodes and form c time intervals
2: $D_{min} \leftarrow \infty$
3: $X \leftarrow \emptyset$
4: **for** $i \leftarrow 1$ to c **do**
5: $x_i \leftarrow$ random active instance within time interval i
6: Augment active instance x_i to working schedule Γ_b of node b
7: Calculate cross-traffic delay D_b using Equation 4
8: **if** $D_b < D_{min}$ **then**
9: $D_b \leftarrow D_{min}$
10: $X \leftarrow x_i$
11: **end if**
12: Remove active instance x_i from working schedule Γ_b of node b
13: **end for**
14: **return** X

it incurs little energy overhead for a node b to perform such procedure in real-time.

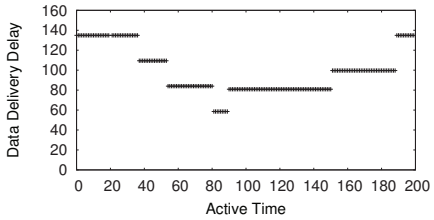


Fig. 10. Stair Effect of Cross-Traffic Delay

To further illustrate Theorem 5.2 and Algorithm 1, we give an example in Figure 10. Assuming one period time contains 200 time instances, Figure 10 shows the expected cross-traffic delay at a node for different augmented active instances. For example, the delay corresponding to active instance 1 represents the delay at node b after augmenting an active instance at time 1. The node shown in Figure 10 has a predecessor with active instances $(36, 53, 80)$ and a successor with active instances $(90, 151, 189)$. According to our analysis above, we can divide one period time into the following intervals: $(36, 53)$, $(53, 80)$, $(80, 90)$, $(90, 151)$, $(151, 189)$, $(189, 36)$.

From Figure 10, it is clear that we have a stair effect of delays at the node among the above time intervals, which is consistent with our analysis. The optimal augmented active instance, therefore, is any value within interval $(80, 90)$.

D. Bursty Augmentation and Decrement

In the previous two subsections, we introduced the optimal solutions for augmenting and decreasing *a single active instance* for the scenario that energy variation is slow. However, the change in harvested energy could be bursty and therefore a node may need to increase or decrease multiple active instances simultaneously. In this section, we present a minimal cost solution for augmenting and decreasing multiple active instances.

A straightforward exhaustive search for multiple active instances augmentation and decrement is no longer acceptable since the computational complexity grows exponentially with the number of augmenting or decreasing active instances. Therefore, a low-cost solution that guarantees optimality of multiple active instances augmentation and decrement is desirable. Fortunately, we observe that multiple active instances augmentation and decrement can be *optimally* solved using a greedy selection.

The main idea of greedy solution is that by applying single active instance augmentation or decrement n times, we can obtain the optimal solution for augmenting or decreasing n active instances at a node. The detailed proof is omitted due to the space constraints. Since the computational complexity of single active instance augmentation or decrement is $\mathcal{O}(1)$, the complexity for augmenting or decrementing n active instances is $\mathcal{O}(n)$.

VI. Maintaining Logical Connectivity

In the previous section, energy synchronization control is designed to adjust the working schedules at receiving nodes, assuming the schedule of predecessors and successors are known and up-to-date. In this section, we investigate the impact of obsolete schedules and how to maintain connectivity while updating schedules.

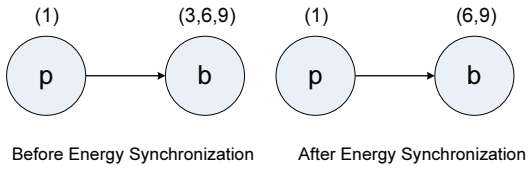


Fig. 11. Impact of Energy Synchronization Example

A. Impact of Schedule Updates

To understand how obsolete schedules can affect the connectivity between nodes, let us start with the example shown in Figure 11, where the working schedule of node b has been changed but has not yet been updated to node p:

- **Unnecessary loss:** If node b decreases its duty-cycle due to insufficient energy supply, then one or more original active instances at node b may be removed. However, before node p is updated, it would continue to deliver packets according to the old (obsolete) working schedule of node b, which could suffer significant greater packet loss than necessary. For example, as shown in Figure 11, node b reduces its duty-cycle by changing its working schedule from $(3, 6, 9)$ to $(6, 9)$. Assuming predecessor node p has a packet ready at time 1, unaware of the new working schedule at node b, node p would try to deliver the packet with active instance sequence $(3, 6, 9, 3, 6, 9, ...)$ until the packet is successfully delivered or the number of retransmissions reaches the bound R_{max}. Since node p would always fail at attempted transmissions at removed active instance 3, the data delivery ratio therefore is clearly decreased.
- **Suboptimal delay:** Similarly, if node b increases its duty-cycle and a predecessor node is unaware of the augmented active instances at node b, it would continue to deliver the packets at node b's original active instances rather than take advantage of the augmented active instances at node b to reduce the delivery delay. In addition, for both predecessors and successors, the new schedule of node b is essential for them to perform effective energy synchronization. Therefore, it is crucial for node b to promptly disseminate its new working schedule to its predecessor and successor nodes.

B. Shuffle-Based vs. Adjustment-Based Energy Synchronization

There are two general approaches that can be taken when node b increases or decreases its duty cycles. Node b can either generate a complete new working schedule with a given new energy budget (termed a *shuffle*), or node b can increase or decrease its duty-cycle on top of its previous working schedule (termed an *adjustment*). For example, assume node b currently has active instances $(3, 6)$ and the current energy supply could afford to increase one more active instance. A shuffle would generate a completely new schedule solely based on the packet ready time at predecessors and active instances at successor nodes. Therefore, the new working schedule at node b after one active instance augmentation could be $(2, 7, 9)$, which has no overlap with the previous schedule. On the other hand, an adjustment would add the augmented active instance to the previous schedule. Consequently, the new working schedule produced by the adjustment could be $(3, 6, 8)$, where $(3, 6)$ are identical to the previous working schedule.

One interesting phenomena of duty-cycled sensor networks is the separation between *physical connectivity* and *logical connectivity*. Two nodes are *physically connected* if they are within each other's communication range and *logically connected* only if they can communicate. Unlike traditional networks, a low-duty-cycle network could be physically connected, but logically partitioned if nodes do not know each other's working schedules.

Let logical connectivity ℓ_{ab} be the packet delivery ratio between two nodes a and b, after R_{max} retransmissions. Let Γ_b be the set of first R_{max} active instances in the original schedule and Γ'_b be the new schedule of node b. The logical connectivity ℓ_{ab}, therefore, is:

$$\ell_{ab} = 1 - (1 - p_{ab})^K \ where \ K = |\Gamma_b \cap \Gamma'_b| \qquad (5)$$

1) Shuffle: In the case of a complete shuffle, $|\Gamma_b \cap \Gamma'_b| = 0$, therefore, logical connectivity $\ell_{ab} = 0$. In other words, node a and b are disconnected. In the case of a partial shuffle, node a experiences the long packet delivery delay as well as low packet delivery ratio before it receives the new schedule from node b. This is because without knowing the new working schedule of node b, node a tries to deliver the data during node b's original active instances. However, most of the original active instances no longer exist due to the shuffling. Consequently, a packet might experience a long delivery delay (after several pointless transmissions), or even get dropped because it exceeded the maximum number of retransmissions.

On the other hand, a shuffle creates an optimal working schedule at node b with the known packet ready times at predecessors and active instances at successors. After the new schedule is updated, a shorter delay is expected.

2) Adjustment: In the case of an adjustment, $|\Gamma_b \cap \Gamma'_b| = \min\{|\Gamma_b|, |\Gamma'_b|\}$, the logical connectivity is optimally maintained. Adjustment preserves the previous schedule when the duty-cycle increases, while it keeps the majority of the previous schedule when the duty-cycle decreases. Consequently, before a predecessor receives the new schedule of node b for the increased duty-cycle, its packet delivery delay and delivery ratio would remain unchanged. Similarly, for the decreased duty-cycle, the packet delivery delay for a predecessor unaware of the new schedule of node b would be consistent with the scenario when the predecessor knows the new schedule since in either case the predecessor would not be able to deliver the data during the removed active instances at node b. The packet delivery ratio, however, would be slightly reduced since the predecessor wastes a certain number of attempted transmissions at the time instance that node b is no longer awake to receive the packet.

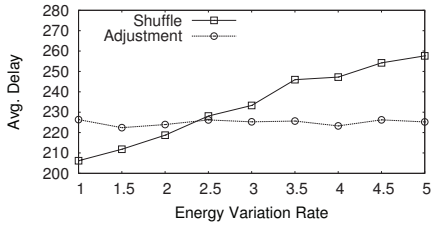


Fig. 12. Delay vs. Energy Variation Rate

3) Comparison: Since adjustment produces smaller delays before the new working schedule reaches predecessor nodes and shuffle creates smaller delays after the new working schedule reaches predecessor nodes, the overall delay at a node therefore is influenced by the energy variation rate at the node. Figure 12 shows average cross-traffic delays for both adjustment and shuffle under different energy variation rates. In Figure 12, the energy variation rate denotes the average number of energy changes over a period of 100,000 units of time. When the energy variation rate is low, the delay after schedule dissemination dominates the overall delay and shuffle therefore has a smaller overall delay than that of adjustment. For example, when the energy variation rate is 1, the overall delay for adjustment and shuffle is 226.34 and 206.17 units of time, respectively. As the energy variation rate becomes larger, the delay before schedule dissemination weights more in overall cross-traffic delay. Consequently, adjustment that has a smaller delay before schedule dissemination also has a smaller overall delay than shuffle. As shown in Figure 12, from energy variation rate 2.5 to 5, adjustment has smaller delay than that of shuffle. At energy variation rate 5, the delay for adjustment and shuffle is 225.22 and 257.65, respectively. This study indicates that the design options on ESC should be decided based on how fast ambient energy changes over time.

VII. Practical Issues

This section completes the description of our ESC design by discussing several practical design issues, such as time synchronization and multiple transmissions within a time instance.

A. Low-cost Time Synchronization

For the sake of clarity, we introduce ESC design in a synchronized mode. Clearly, the operation of ESC depends on neither neighbor time instance nor global synchronization. It is sufficient for ESC only knows the *wake-up time interval* of predecessor and successor nodes. To know those wake-up time intervals, simple and low-cost local synchronization techniques [26] can achieve an accuracy of $2.24\mu s$ with the cost of exchange a few bytes of packets among neighboring nodes every 15 minutes. Since an active instance is typically ranges from $2000\mu s$ to $20{,}000\mu s$, the accuracy of $2.24\mu s$ is far more than sufficient. In addition, ESC does not require the transmission starts at the beginning of an active instance, which further relaxes the requirement of accuracy for time synchronization.

B. Multiple Transmissions within a Time Instance

While describing our network model in Section III, we assume there is at most one packet transmission during an active instance. This is true if nodes are equipped with slow radio. However, if fast radio are used, it is possible to transmit multiple packets within an active instance. To accommodate such scenario in modeling of cross-traffic delay, we can simply rewrite the bidirectional link quality between two nodes as $p'_{ab} = 1 - (1 - p_{ab})^m$, where m is the maximum number of transmissions allowed in an active instance. Essentially, the new p'_{ab} represents the probability that the receiver received the packet by m transmissions.

VIII. Implementation and Evaluation

In order to validate the performance and feasibility of ESC in practice, we have fully implemented ESC on the TinyOS/Mote platform in nesC. To compare the performance of ESC, we also implemented a random working schedule synchronization scheme that randomly adjusts active instances of the original node working schedule with increasing or decreasing node duty cycles.

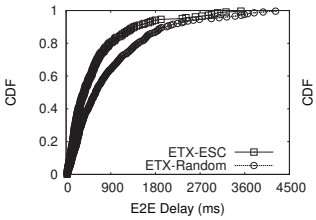


Fig. 13. ETX E2E Delay

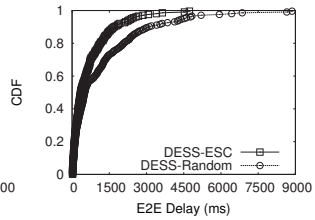


Fig. 14. DESS E2E Delay

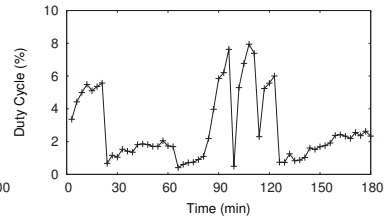


Fig. 15. Node Duty-Cycle Over Time

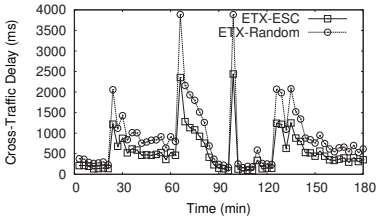


Fig. 16. Cross-Traffic Delay Over Time

A. Experiment Setup

During the experiment, we randomly placed 30 MicaZ nodes on our test-bed. The transmission power at MicaZ motes is tuned down to form a multi-hop network. Neighboring nodes are synchronized using FTSP [26] and each active instance lasts for $20ms$. The available energy budget over time at each node is derived from the actual energy harvesting rate measured at our running prototype and the corresponding energy harvesting model [16]. The range of duty cycle varies from around 0.2% to 10%. According to the available energy budget, each node turns on and off its radio based on the energy synchronized working schedule. To study the performance of ESC under various routing protocols, we also implemented link-quality-based ETX [27] and sleep-latency-based DESS [23] on motes.

B. Performance Comparison

In this section, we compare the E2E delay for both ESC and the randomized scheme. During the experiment, for each routing protocol, over 1000 packets are transmitted from a random source to a random destination. To ensure the fair comparison, ESC packets and randomized energy synchronization packets are sent alternatively to minimize the impact of energy variation and link quality fluctuation.

In Figure 13 we study the E2E delay for ESC and the randomized scheme under ETX. Clearly, ESC performs much better than the randomized scheme. While 80% of ESC packets reach their destinations within 877ms, the corresponding percentile for the randomized scheme is 1451ms, which is about 65% increase. Since ETX picks the route with the minimum number of expected transmissions, the performance gap between ESC and the randomized scheme therefore is mainly due to the minimized cross-traffic delay for ESC.

Similar to the results for ETX, ESC also significantly outperforms the randomized scheme under DESS. While the 80% percentile for ESC is 1131ms, the corresponding number for the randomized scheme is 2091ms, which almost doubles the number of ESC. In addition, the randomized scheme has much longer tail than ESC. While all messages for ESC reach the destinations within 6299ms, the longest E2E delay for the randomized scheme is 12343ms. The reason for such long tail of the randomized scheme is because the penalty of a failed transmission for the randomized scheme is much larger than the ESC, as ESC has carefully scheduled the radio activity to minimize the impact of the failed transmissions.

To further reveal the performance of ESC over time dimension, Figure 15 and Figure 16 show the duty-cycle of a deployed node and its corresponding cross-traffic delay under ESC and the randomized scheme over a period of 3 hours. By comparing Figure 15 and Figure 16, we can see the cross-traffic delay matches the available duty-cycle well. For example, the peaks of delay occur when the node duty-cycle drops to around 0.4% at time 65 and 100. In addition, although both ESC and the randomized scheme react to the duty-cycle change promptly, the cross-traffic delay for ESC is always smaller than that of the randomized scheme. This consistent smaller cross-traffic delay of ESC over time further explains the smaller E2E delay for ESC in Figure 13.

IX. Simulation Evaluation

In addition to test-bed evaluation in Section VIII, to understand the system performance of ESC under numerous network settings, in this section we provide simulation results with over 1000 sensor nodes. To investigate the flexibility of ESC design, we choose two state-of-the-art solutions as underlying routing protocols:

- Link-Quality-based: ETX [27] in MobiCom'03
- Sleep-Latency-based: DESS [23] in INFOCOM'05

A. Simulation Setup

In the simulation, except where otherwise specified, we deploy up to 1,200 sensor nodes randomly in a 400m$\times$400m square field. A sink is positioned in the center of the deployment field, and each sensor node sends its packet to the sink over multiple hops. The radio model was implemented according to [28], which considers the oscillation nature of the radio links and has several adjustable parameters. During the simulation, we set these parameters strictly according to the CC2420 radio hardware specification [29].

Each experiment was repeated 100 times with different random seeds, node deployments, and node working schedules. Data collected at each node were obtained by averaging over 10000 source-to-sink communications. The 95% confidence intervals are within 1$\sim$4% of the means.

B. System Performance Over Time

In this section, we reveal the effectiveness of ESC over time in terms of communication delays. Figure 20 shows the average cross-traffic delay at a node over a period of 25,000 units of time. At time 0, active instances are allocated randomly within nodes (hence not optimally). The node increases its duty-cycle at time 5,000 and 10,000 and decreases its duty-cycle at time 15,000 and 20,000. It is clear that after the node increases its duty-cycle at time 5,000, the delay at the node

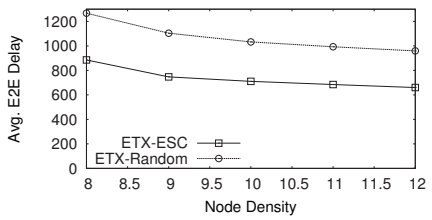


Fig. 17. ETX Delay vs. Node Density

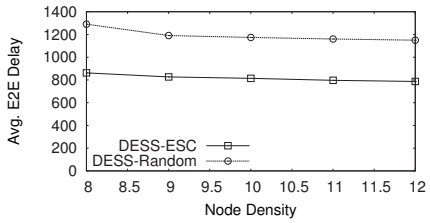


Fig. 18. DESS Delay vs. Node Density

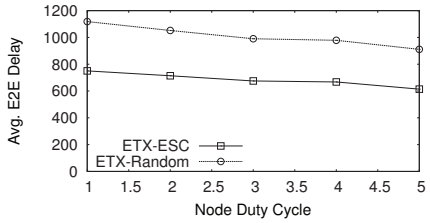


Fig. 19. ETX Delay vs. Duty Cycle

significantly drops. For example, within time interval $[0, 5000]$, the average delay is 249.26 units of time. In contrast, during time $[5000, 10000]$, the average delay drops to 90.51, which is around only 36.3% of the original delay. After increasing the duty-cycle again at time 10,000, the delay at the node further reduces to 50.32 during time $[10000, 15000]$, almost half the previous delay. When the duty-cycle decreases at time 15,000, the average delay only slightly increases to 51.36 units of time within time interval $[15000, 20000]$. Finally, when we further reduce the duty-cycle at time 20,000, the delay increases to 113.66 which is only around 45.6% of the initial delay during time $[0, 5000]$, when allocation is not optimal. From this figure, it is clear that ESC effectively reduces the delay at the node when its duty-cycle increases while it minimally increases the delay when the node decreases its duty-cycle, converging gradually from an initial not-optimal allocation into an optimal allocation.

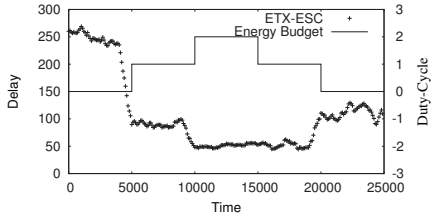


Fig. 20. Delay Over Time

C. Impact of Node Densities

In this section, we examine the impact of node densities on E2E delay for both ETX and DESS networks with varying energy supplies over time.

During the simulation, for each node both ESC and the randomized energy synchronization scheme described in Section VIII have the same energy supply over time to ensure fair comparisons. As can clearly be seen from both Figure 17 and Figure 18, ESC has a much smaller delay than the Randomized scheme at all node densities for both ETX and DESS. For example, at node density 10, ETX-ESC has a delay of 710.64 units of time while ETX-Random has a delay of 1032.51 units of time, which is about 45% larger than the delay for ETX-ESC. Similarly for DESS at node density 10, the delay for DESS-ESC and DESS-Random is 814.02 units of time and 1173.95 units of time, respectively.

D. Impact of Node Duty Cycles

In this section, we study the impact of node duty cycles on E2E delay for applying ESC to ETX and DESS in energy-varying sustainable sensor networks. In both Figure 19 and Figure 21, we can see that ESC outperforms randomized scheme at all node duty cycles. As the node duty cycle increases, E2E delays for both ETX and DESS under ESC and randomized scheme are decreasing. This is because with a higher node duty-cycle in the network, the sleep latency between a sender and a receiver is reduced, as there are more active instances at the receiver to receive incoming packets from sending nodes. For example, average E2E delays for ESC-ETX decreases from 750.27 units of time to 614.15 units time while the delays for Random-ETX decreases from 1118.48 to 911.31 units of time.

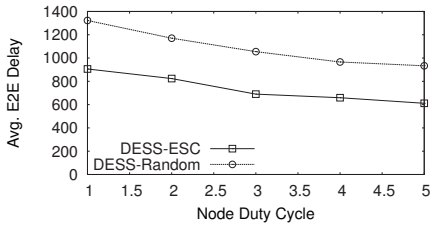


Fig. 21. DESS Delay vs. Duty Cycle

X. Related Work

Several technologies have been developed to extract energy from the environment, including solar, motion, biochemical and vibrational [11], [13]. Building on those energy-harvesting technologies, researchers have designed various types of platforms to collect ambient energy from the environment with optimal efficiency [12], [17], [30]. To fully utilize the harvested energy and ensure the sustainable operation of sensor node, Kansal et al. [15], [24] and Vigorito et al. [25] have presented both theoretical and experimental results on deciding the appropriate working duty-cycle of sensor nodes with information on current energy harvesting rates. To further study the impact of energy leakage for energy-harvesting sensor networks, TwinStar system suggests node duty-cycle base on user specified lifetime and energy information including energy harvesting rate, remaining energy in the system and energy leakage rate [16].

On the other hand, with the growing gap between energy supply and demand recently, there has been a surge of interest on intermittently connected networks. For scenarios with mobile nodes, a number of effective solutions has been proposed to data communication in such networks [31]–[33]. For scenarios with low-duty-cycle nodes, by assuming perfect link qualities, both work [23], [34] introduce serval techniques for

minimizing communication latency while providing energy-efficient periodic node working schedules. To address both low-duty-cycle and unreliable communication links, Gu et al. [18] introduce a dynamic switch-based forwarding using optimized forwarding sequences. Su et al. [21] propose both on-demand and proactive algorithms for routing packets in low-duty-cycle networks. More recently, serval efficient flooding protocols have been introduced to tackle the unique challenge in low-duty-cycle sensor networks [35], [36].

However, none of those prior works investigate how changing the duty-cycle of sensor nodes affects communication performance in sustainable sensor networks and how we can adaptively synchronize node working schedules with specified duty-cycle budgets. In this work, we advance state-of-the-art solutions for both energy-harvesting and low-duty-cycle sensor networks and providing effective methods of synchronizing node working schedules with varying duty-cycle budgets.

XI. Conclusion

In this work, we reveal that cross-traffic delay through a duty-cycled node is determined only by the number of active instances at intervals, partitioned by active instances of predecessor and successor nodes. This allows us to design energy-synchronized control with $\mathcal{O}(1)$ time complexity for sustainable networks in which energy supplies and demands are in balance. In a low-duty-cycle network, updating neighbors' working schedules would be slow, leading to inconsistent views on active instances. To address this issue, we investigate the impact of obsolete working schedules on logical link quality and demonstrate the tradeoff between shuffle-based and adjustment-based allocation under different energy variation rates. Our evaluation demonstrates that ESC can effectively reduce delay and increase delivery ratios, while synchronizing radio activity with available energy.

Acknowledgement

This work was supported in part by NSF grants CNS-0626609, CNS-0626614 and CNS-0917097. We would like to thank the anonymous reviewers and our paper shepherd Ahmed Helmy for the invaluable comments that significantly improved this work.

References

[1] A. Arora *et al.*, "Exscal: Elements of an extreme scale wireless sensor network," in *RTCSA*, 2005.

[2] J. Jeong, Y. Gu, T. He, and D. Du, "VISA: Virtual Scanning Algorithm for Dynamic Protection of Road Networks," in *Proc. of the 28th Annual IEEE Conference on Computer Communications (INFOCOM '09)*, 2009.

[3] H. Morcos, A. Bestavros, and I. Matta, "Amorphous placement and informed diffusion for timely field monitoring by autonomous, resource-constrained, mobile sensors," in *SECON*, 2008.

[4] J. Ko, T. Gao, and A. Terzis, "Empirical study of a medical sensor application in an urban emergency department," in *BodyNets*, 2009.

[5] J. Wu, S. Yang, and M. Cardei, "On maintaining sensor-actor connectivity in wireless sensor and actor networks," in *INFOCOM*, 2008.

[6] O. Younis, M. Krunz, and S. Ramasubramanian, "Coverage without location information," in *ICNP*, 2007.

[7] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. Smith, "S4: Small state and small stretch routing protocol for large wireless sensor networks," in *NSDI*, 2007.

[8] C. Gui and P. Mohapatra, "Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks," in *MobiCom'04*, 2004.

[9] A. T. Hoang and M. Motani, "Collaborative broadcasting and compression in cluster-based wireless sensor networks," *ACM TOSN*, vol. 3, no. 3, 2007.

[10] M. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy, "Efficient geographic routing over lossy links in wireless sensor networks," *ACM TOSN*, vol. 4, no. 3, 2008.

[11] S. Meninger, J. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang, "Vibration-to-electric Energy Conversion," in *ISLPED*, 1999.

[12] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann, and D. Estrin, "Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network," in *ICRA'03*, 2003.

[13] S. Wright, D. Scott, J. Haddow, and M. Rosen, "The Upper Limit to Solar Energy Conversion," in *IECEC*, 2000.

[14] C. Park and P. Chou, "AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes," in *SECON'06*, 2006.

[15] A. Kansal, D. Potter, and M. B. Srivastava, "Performance Aware Tasking for Environmentally Powered Sensor Networks," in *SIGMETRICS '04*, 2004.

[16] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z.-L. Zhang, "Leakage-Aware Energy Synchronization for Wireless Sensor Networks," in *MobiSys '09*, 2009.

[17] X. Jiang, J. Polastre and D. Culler, "Perpetual Environmentally Powered Sensor Networks," in *IPSN'05*, 2005.

[18] Y. Gu and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," in *SenSys '07*, 2007.

[19] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance In Dense Wireless Sensor Networks," in *Sensys 03*, 2003.

[20] Y. Wu, S. Fahmy, and N. B. Shroff, "Energy Efficient Sleep/Wake Scheduling for Multi-Hop Sensor Networks: Non-Convexity and Approximation Algorithm," in *INFOCOM*, 2007.

[21] L. Su, C. Liu, H. Song, and G. Cao, "Routing in intermittently connected sensor networks," in *ICNP*, 2008.

[22] S. He, J. Chen, D. Yau, H. Shao, and Y. Sun, "Energy-efficient capture of stochastic events by global- and local-periodic network coverage," in *MobiHoc*, 2009.

[23] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay Efficient Sleep Scheduling in WIreless Sensor Networks," in *INFOCOM'05*, 2005.

[24] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," *TECS*, vol. 6, no. 4, 2007.

[25] C. Vigorito, D. Ganesan, and A. Bartoeeee, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *SECON'07*, 2007.

[26] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," in *SenSys'04*, 2004.

[27] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High Throughput Path Metric for Multi-Hop Wireless Routing," in *MOBICOM'03*, 2003.

[28] M. Zuniga and B. Krishnamachari, "Analyzing the Transitional Region in Low Power Wireless Links," in *IEEE SECON'04*, 2004.

[29] *CC2420 Product Information and Data Sheet*, Chipcon, avaiable at http://www.chipcon.com/.

[30] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments," in *IPSN'06*, 2006.

[31] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *LNCS*, pp. 239–254, 2004.

[32] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM TON*, vol. 16, no. 1, pp. 77–90, 2008.

[33] M. Musolesi, S. Hailes, and C. Mascolo, "Adaptive routing for intermittently connected mobile ad hoc networks," in *WoWMoM'05*, 2005.

[34] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup Scheduling in Wireless Sensor Networks," in *MobiHoc*, 2006.

[35] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links," in *MobiCom '09*, 2009.

[36] F. Wang and J. Liu, "Duty-cycle-aware broadcast in wireless sensor networks," in *INFOCOM'09*, 2009.